



# Compatible natural gradient policy search

Joni Pajarinen<sup>1,4</sup> · Hong Linh Thai<sup>1</sup> · Riad Akrou<sup>1</sup> · Jan Peters<sup>1,2</sup> ·  
Gerhard Neumann<sup>3</sup>

Received: 21 January 2019 / Accepted: 2 May 2019  
© The Author(s) 2019

## Abstract

Trust-region methods have yielded state-of-the-art results in policy search. A common approach is to use KL-divergence to bound the region of trust resulting in a natural gradient policy update. We show that the natural gradient and trust region optimization are equivalent if we use the natural parameterization of a standard exponential policy distribution in combination with compatible value function approximation. Moreover, we show that standard natural gradient updates may reduce the entropy of the policy according to a wrong schedule leading to premature convergence. To control entropy reduction we introduce a new policy search method called compatible policy search (COPOS) which bounds entropy loss. The experimental results show that COPOS yields state-of-the-art results in challenging continuous control tasks and in discrete partially observable tasks.

**Keywords** Reinforcement learning · Policy search

## 1 Introduction

The natural gradient (Amari 1998) is an integral part of many reinforcement learning (Kakade 2001; Bagnell and Schneider 2003; Peters and Schaal 2008; Geist and Pietquin 2010) and optimization (Wierstra et al. 2008) algorithms. Due to the natural gradient, gradient updates become invariant to affine transformations of the parameter space and the natural gradient is also often used to define a trust-region for the policy update. The trust-region is defined by a bound of the Kullback–Leibler (KL) (Peters et al. 2010; Schulman et al. 2015) divergence between new and old policy and it is well known that the Fisher information matrix, used to compute the natural gradient is a second order approximation of the KL divergence. Such

---

Editors: Karsten Borgwardt, Po-Ling Loh, Evimaria Terzi, Antti Ukkonen.

---

✉ Joni Pajarinen  
[pajarinen@ias.tu-darmstadt.de](mailto:pajarinen@ias.tu-darmstadt.de)

<sup>1</sup> Intelligent Autonomous Systems, TU Darmstadt, Darmstadt, Germany

<sup>2</sup> MPI for Intelligent Systems, Tuebingen, Germany

<sup>3</sup> Lincoln Center for Autonomous Systems, University of Lincoln, Lincoln, UK

<sup>4</sup> Tampere University, Tampere, Finland

trust-region optimization is common in policy search and has been successfully used to optimize neural network policies.

However, many properties of the natural gradient are still under-explored, such as compatible value function approximation (Sutton et al. 1999) for neural networks, the approximation quality of the KL-divergence and the online performance of the natural gradient. We analyze the convergence of the natural gradient analytically and empirically and show that the natural gradient does not give fast convergence properties if we do not add an entropy regularization term. This entropy regularization term results in a new update rule which ensures that the policy loses entropy at the correct pace, leading to convergence to a good policy. We further show that the natural gradient is the optimal (and not the approximate) solution to a trust region optimization problem for log-linear models if the natural parameters of the distribution are optimized and we use compatible value function approximation.

We analyze compatible value function approximation for neural networks and show that the components of this approximation are composed of two terms, a state value function which is subtracted from a state-action value function. While it is well known that the compatible function approximation denotes an advantage function, the exact structure was unclear. We show that using compatible value function approximation, we can derive similar algorithms to Trust Region Policy Search that obtain the policy update in closed form. A summary of our contributions is as follows:

- It is well known that the second-order Taylor approximation to trust-region optimization with a KL-divergence bound leads to an update direction identical to the natural gradient. However, what is not known is that when using the natural parameterization for an exponential policy and using compatible features we can compute the step-size for the natural gradient that solves the trust-region update exactly for the log-linear parameters.
- When using an entropy bound in addition to the common KL-divergence bound, the compatible features allow us to compute the exact update for the trust-region problem in the log-linear case and for a state independent covariance also in the non-linear case.
- Our new algorithm called Compatible Policy Search (COPOS), based on the above insights, outperforms comparison methods in both continuous control and partially observable discrete action experiments due to entropy control allowing for principled exploration.

## 2 Preliminaries

This section discusses background information needed to understand our compatible policy search approach. We first go into Markov decision process (MDP) basics and introduce the optimization objective. We continue by showing how trust region methods can help with challenges in updating the policy by using a KL-divergence bound, continue with the classic policy gradient update, introduce the natural gradient and the connection to the KL-divergence bound. Moreover, we introduce the compatible value function approximation and connect it to the natural gradient. Finally, this section concludes by showing how the optimization problem resulting from using an entropy bound to control exploration can be solved.

Following standard notation, we denote an infinite-horizon discounted Markov decision process (MDP) by the tuple  $(\mathcal{S}, \mathcal{A}, p, r, p_0, \gamma)$ , where  $\mathcal{S}$  is a finite set of states and  $\mathcal{A}$  is a finite set of actions.  $p(s_{t+1}|s_t, a_t)$  denotes the probability of moving from state  $s_t$  to  $s_{t+1}$  when the agent executes action  $a_t$  at time step  $t$ . We assume  $p(s_{t+1}|s_t, a_t)$  is stationary and unknown but that we can sample from  $p(s_{t+1}|s_t, a_t)$  either in simulation or from a physical

system.  $p_0(s)$  denotes the initial state distribution,  $\gamma \in (0, 1)$  the discount factor, and  $r(s_t, a_t)$  denotes the real valued reward in state  $s_t$  when agent executes action  $a_t$ . The goal is to find a stationary policy  $\pi(a_t|s_t)$  that maximizes the expected reward  $\mathbb{E}_{s_0, a_0, \dots} [\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)]$ , where  $s_0 \sim p_0(s)$ ,  $s_{t+1} \sim p(s_{t+1}|s_t, a_t)$  and  $a_t \sim \pi(a_t|s_t)$ . In the following we will use the notation for the continuous case, where  $\mathcal{S}$  and  $\mathcal{A}$  denote finite dimensional real valued vector spaces and  $s$  denotes the real-valued state vector and  $\mathbf{a}$  the real-valued action vector. For discrete states and actions integrals can be replaced in the following by sums.

The expected reward can be defined as (Schulman et al. 2015)

$$J(\pi) = \iint p_{\pi}(s) \pi(\mathbf{a}|s) Q^{\pi}(s, \mathbf{a}) ds d\mathbf{a}, \quad (1)$$

where  $p_{\pi}(s)$  denotes a (discounted) state distribution induced by policy  $\pi$  and

$$\begin{aligned} Q^{\pi}(s_t, \mathbf{a}_t) &= \mathbb{E}_{s_{t+1}, a_{t+1}, \dots} [\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)], \\ V^{\pi}(s_t) &= \mathbb{E}_{a_t, s_{t+1}, \dots} [\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)], \quad A^{\pi}(s_t, \mathbf{a}_t) = Q^{\pi}(s_t, \mathbf{a}_t) - V^{\pi}(s_t) \end{aligned}$$

denote the state-action value function  $Q^{\pi}(s_t, \mathbf{a}_t)$ , value function  $V^{\pi}(s_t)$ , and advantage function  $A^{\pi}(s_t, \mathbf{a}_t)$ .

The goal in policy search is to find a policy  $\pi(\mathbf{a}|s)$  that maximizes Eq. (1). Usually, policy search computes in each iteration a new improved policy  $\pi$  based on samples generated using the old policy  $\pi_{\text{old}}$  since maximizing Eq. (1) directly is too challenging. However, since the estimates for  $p_{\pi}(s)$  and  $Q^{\pi}(s, \mathbf{a})$  are based on the old policy, that is,  $p_{\pi_{\text{old}}}(s)$  and  $Q^{\pi_{\text{old}}}(s, \mathbf{a})$  are actually used in Eq. (1), they may not be valid for the new policy. A solution to this is to use Trust-Region Optimization methods which keep the new policy sufficiently close to the old policy. Trust-Region Optimization for policy search was first introduced in the relative entropy policy search (REPS) algorithm (Peters et al. 2010). Many variants of this algorithm exist (Akrouir et al. 2016; Abdolmaleki et al. 2015; Daniel et al. 2016; Akrouir et al. 2018). All these algorithms use a bound for the KL-divergence of the policy update which prevents the policy update from being unstable as the new policy will not go too far away from areas it has not seen before. Moreover, the bound prevents the policy from being too greedy. Trust region policy optimization (TRPO) (Schulman et al. 2015) uses this bound to optimize neural network policies. The policy update can be formulated as finding a policy that maximizes the objective in Eq. (1) under the KL-constraint:

$$\operatorname{argmax}_{\pi} \mathbb{E}_{p_{\pi_{\text{old}}}(s)} \left[ \int \pi(\mathbf{a}|s) Q^{\pi_{\text{old}}}(s, \mathbf{a}) d\mathbf{a} \right] \quad (2)$$

$$\text{s.t. } \mathbb{E}_{p_{\pi_{\text{old}}}(s)} [\text{KL}(\pi(\cdot|s) || \pi_{\text{old}}(\cdot|s))] \leq \epsilon, \quad (3)$$

where  $Q^{\pi_{\text{old}}}(s, \mathbf{a})$  denotes the future accumulated reward values of the old policy  $\pi_{\text{old}}$ ,  $p_{\pi_{\text{old}}}(s)$  the (discounted) state distribution under the old policy, and  $\epsilon$  is a constant hyper-parameter. For an  $\epsilon$  small enough the state-value and state distribution estimates generated using the old policy are valid also for the new policy since the new and old policy are sufficiently close to each other.

**Policy gradient** We consider parameterized policies  $\pi_{\theta}(\mathbf{a}|s)$  with parameter vector  $\theta$ . A policy can be improved by modifying the policy parameters in the direction of the policy gradient which is computed w.r.t. Eq. (1). The “vanilla” policy gradient (Williams 1992; Sutton et al. 1999) obtained by the likelihood ratio trick is given by

$$\begin{aligned}\nabla_{\theta} J_{\text{PG}} &= \iint p(s) \pi_{\theta}(a|s) \nabla \log \pi_{\theta}(a|s) Q^{\pi}(s, a) ds da \\ &\approx \sum_i \nabla \log \pi_{\theta}(a_i|s_i) Q^{\pi_{\text{old}}}(s_i, a_i).\end{aligned}$$

The Q-values can be computed by Monte-Carlo estimates (high variance), that is,  $Q^{\pi_{\text{old}}}(s_i, a_i) \approx \sum_{h=0}^{\infty} \gamma^h r_{t+h}$  or estimated by policy evaluation techniques (typically high bias). We can further subtract a state-dependent baseline  $V(s)$  which decreases the variance of the gradient estimate while leaving it unbiased, that is,

$$\nabla_{\theta} J_{\text{PG}} \approx \sum_i \nabla \log \pi_{\theta}(a_i|s_i) (Q^{\pi_{\text{old}}}(s_i, a_i) - V(s_i)).$$

*Natural gradient* Contrary to the “vanilla” policy gradient, the natural gradient (Amari 1998) method uses the steepest descent direction in a Riemannian manifold, so it is effective in learning, avoiding plateaus. The natural gradient can be obtained by using a second order Taylor approximation for the KL divergence, that is,  $\mathbb{E}_{p(s)} [\text{KL}(\pi_{\theta+\alpha}(\cdot|s) || \pi_{\theta}(\cdot|s))] \approx \alpha^T \mathbf{F} \alpha$ , where  $\mathbf{F}$  is the Fisher information matrix (Amari 1998). The natural gradient is now defined as the update direction that is most correlated with the standard policy gradient and has a limited approximate KL, that is,

$$\nabla_{\theta} J_{\text{NAC}} = \operatorname{argmax}_{\alpha} \alpha^T \nabla_{\theta} J_{\text{PG}} \quad \text{s.t.} \quad \alpha^T \mathbf{F} \alpha \leq \epsilon$$

resulting in

$$\nabla_{\theta} J_{\text{NAC}} = \eta^{-1} \mathbf{F}^{-1} \nabla_{\theta} J_{\text{PG}},$$

where  $\eta$  is a Lagrange multiplier.

*Compatible value function approximation* It is well known that we can obtain an unbiased gradient with typically smaller variance if compatible value function approximation is used (Sutton et al. 1999). An approximation of the Monte-Carlo estimates  $\tilde{G}_{\mathbf{w}}^{\pi_{\text{old}}}(s, a) = \phi(s, a)^T \mathbf{w}$  is compatible to the policy  $\pi_{\theta}(a|s)$ , if the features  $\phi(s, a)$  are given by the log gradients of the policy, that is,  $\phi(s, a) = \nabla_{\theta} \log \pi_{\theta}(a|s)$ . The parameter  $\mathbf{w}$  of the approximation  $\tilde{G}_{\mathbf{w}}^{\pi_{\text{old}}}(s, a)$  is the solution of the least squares problem

$$\mathbf{w}^* = \operatorname{argmin}_{\mathbf{w}} \mathbb{E}_{p(s) \pi_{\theta}(a|s)} \left[ \left( Q^{\pi_{\text{old}}}(s, a) - \phi(s, a)^T \mathbf{w} \right)^2 \right].$$

Peters and Schaal (2008) showed that in the case of compatible value function approximation, the inverse of the Fisher information matrix cancels with the matrix spanned by the compatible features and, hence,  $\nabla_{\theta} J_{\text{NAC}} = \eta^{-1} \mathbf{w}^*$ . Another interesting observation is that the compatible value function approximation is in fact not an approximation for the Q-function but for the advantage function  $A^{\pi_{\text{old}}}(s, a) = Q^{\pi_{\text{old}}}(s, a) - V^{\pi_{\text{old}}}(s)$  as the compatible features are always zero mean. In Sect. 3, we show how with compatible value function approximation the natural gradient directly gives us an exact solution to the trust region optimization problem instead of requiring a search for the update step size to satisfy the KL-divergence bound in trust region optimization.

**Entropy regularization** Recently, some approaches (Abdolmaleki et al. 2015; Akrouer et al. 2016; Mnih et al. 2016; O’Donoghue et al. 2016) use an additional bound for the entropy of the resulting policy. The entropy bound can be beneficial since it allows to limit the change in exploration potentially preventing greedy policy convergence. The trust region problem is in this case given by

$$\begin{aligned}
& \operatorname{argmax}_{\pi} \mathbb{E}_{p_{\text{old}}(s)} \left[ \int \pi(a|s) Q^{\pi_{\text{old}}}(s, a) da \right] \\
\text{s.t. } & \mathbb{E}_{p_{\text{old}}(s)} [\text{KL}(\pi(\cdot|s) || \pi_{\text{old}}(\cdot|s))] \leq \epsilon \\
& \mathbb{E}_{p_{\text{old}}(s)} [H(\pi_{\text{old}}(\cdot|s)) - H(\pi(\cdot|s))] \leq \beta,
\end{aligned} \tag{4}$$

where the second constraint limits the expected loss in entropy ( $H(\cdot)$  denotes Shannon entropy in the discrete case and differential entropy in the continuous case) for the new distribution [applying an entropy constraint only on  $\pi(a|s)$  but adjusting  $\beta$  according to  $\pi_{\text{old}}(a|s)$  is equivalent in (Akrou et al. 2016, 2018)]. The policy update rule can be formed for the constrained optimization problem by using the method of Lagrange multipliers:

$$\pi(a|s) \propto \pi_{\text{old}}(a|s)^{\frac{\eta}{\eta+\omega}} \exp\left(\frac{Q^{\pi_{\text{old}}}(s, a)}{\eta + \omega}\right), \tag{5}$$

where  $\eta$  and  $\omega$  are Lagrange multipliers (Akrou et al. 2016).  $\eta$  is associated with the KL-divergence bound  $\epsilon$  and  $\omega$  is related to the entropy bound  $\beta$ . Note that, for  $\omega = 0$ , the entropy bound is not active and therefore, the solution is equivalent to the standard trust region solution. It has been realized that the entropy bound is needed to prevent premature convergence issues connected with the natural gradient. We show that these premature convergence issues are inherent to the natural gradient as it always reduces entropy of the distribution. In contrast, entropy control can prevent this.

### 3 Compatible policy search with natural parameters

In this section, we analyze the natural gradient update equations for exponential family distributions. This analysis reveals an important connection between the natural gradient and the trust region optimization: Both are equivalent if we use the natural parameterization of the distribution in combination with compatible value function approximation. This is an important insight as the natural gradient now provides the optimal solution for a given trust region, not just an approximation which is commonly believed: for example, Schulman et al. (2015) have to use line search to fit the natural gradient update to the KL-divergence bound. Moreover, this insight can be applied together with the entropy bound to control policy exploration and get a closed form update in the case of compatible log-linear policies. Furthermore, the use of compatible value function approximation has several advantages in terms of variance reduction which can not be achieved with the plain Monte-Carlo estimates which we leave for future work. We also present an analysis of the online performance of the natural gradient and show that entropy regularization can converge exponentially faster. Finally, we present our new algorithm for Compatible Policy Search (COPOS) which uses the insights above.

#### 3.1 Equivalence of natural gradients and trust region optimization

We first consider soft-max distributions that are log-linear in the parameters (for example, Gaussian distributions or the Boltzmann distribution) and subsequently extend our results to non-linear soft-max distributions, for example given by neural networks. A log-linear soft-max distribution can be represented as

$$\pi(a|s) = \frac{\exp(\psi(s, a)^T \theta)}{\int \exp(\psi(s, a)^T \theta) da}.$$

Note that also Gaussian distributions can be represented this way [see, for example, Eq. (9)], however, the natural parameterization is commonly not used for Gaussian distributions. Typically, the Gaussian is parameterized by the mean  $\mu$  and the covariance matrix  $\Sigma$ . However, the natural parameterization and our analysis suggest that the precision matrix  $B = \Sigma^{-1}$  and the linear vector  $b = \Sigma^{-1}\mu$  should be used to benefit from many beneficial properties of the natural gradient.

It makes sense to study the exact form of the compatible approximation for these log-linear models. The compatible features are given by

$$\phi(s, a) = \nabla_{\theta} \log \pi_{\theta}(a|s) = \psi(s, a) - \mathbb{E}_{\pi(\cdot|s)} [\psi(s, \cdot)].$$

As we can see, the compatible feature space is always zero mean, which is inline with the observation that the compatible approximation  $\tilde{G}_w^{\pi_{\text{old}}}(s, a)$  is an advantage function. Moreover, the structure of the features suggests that the advantage function is composed of a term for the Q-function  $\tilde{Q}_w(s, a) = \psi(s, a)^T w$  and for the value function  $\tilde{V}_w(s) = \mathbb{E}_{\pi(\cdot|s)} [\tilde{Q}_w(s, a)]$ , that is,

$$\tilde{G}_w^{\pi_{\text{old}}}(s, a) = \psi(s, a)^T w - \mathbb{E}_{\pi(\cdot|s)} [\psi(s, \cdot)^T w]$$

We can now directly use the compatible advantage function  $\tilde{G}_w^{\pi_{\text{old}}}(s, a)$  in our trust region optimization problem given in Eq. (3). The resulting policy is then given by

$$\begin{aligned} \pi(a|s) &\propto \pi_{\text{old}}(a|s) \exp \left( \frac{\psi(s, a)^T w - \mathbb{E}_{\pi(\cdot|s)} [\psi(s, \cdot)^T w]}{\eta} \right) \\ &\propto \exp (\psi(s, a)^T (\theta_{\text{old}} + \eta^{-1} w)). \end{aligned}$$

Note that the value function part of  $\tilde{G}_w$  does not influence the updated policy. Hence, if we use the natural parameterization of the distributions in combination with compatible function approximation, then we directly get a parametric update of the form

$$\theta = \theta_{\text{old}} + \eta^{-1} w. \quad (6)$$

Furthermore, the suggested update is equivalent to the natural gradient update: The natural gradient is the *optimal solution* for a given trust region problem and not just an approximation. However, this statement only holds if we use natural parameters and compatible value function approximation. Moreover, the update needs only the Q-function part of the compatible function approximation.

We can do a similar analysis for the optimization problem with entropy regularization given in Eq. (4) using Eq. (5). The optimal policy given the compatible value function approximation is now given by

$$\pi(a|s) \propto \exp \left( \psi(s, a)^T \left( \frac{\eta \theta_{\text{old}} + w}{\eta + \omega} \right) \right), \quad \text{yielding} \quad \theta = \frac{\eta \theta_{\text{old}} + w}{\eta + \omega}. \quad (7)$$

In comparison to the standard natural gradient, the influence of the old parameter vector is diminished by the factor  $\eta/(\eta + \omega)$  which will play an important role for our further analysis.

### 3.2 Compatible approximation for neural networks

So far, we have only considered models that are log-linear in the parameters (ignoring the normalization constant). For more complex models, we need to introduce non-linear param-

eters  $\beta$  for the feature vector, that is,  $\psi(s, a) = \psi_\beta(s, a)$ . We are in particular interested in Gaussian policies in the continuous case and softmax policies in the discrete case as they are the standard for continuous and discrete actions, respectively. In the continuous case, we could either use a Gaussian with a constant variance where the mean is parameterized by a neural network, a non-linear interpolation linear feedback controllers with Gaussian noise or also Gaussians with state-dependent variance. For simplicity, we will focus on Gaussians with a constant covariance  $\Sigma$  where the mean is a product of neural network (or any other non-linear function) features  $\varphi_i(s)$  and a mixing matrix  $K$  that could be part of the neural network output layer. The policy and the log policy are then

$$\pi(a|s) = \mathcal{N}\left(a \middle| \sum_i \varphi_i(s) k_i, \Sigma\right) \quad (8)$$

$$\begin{aligned} \log \pi(a|s) &= -0.5 \varphi(s)^T K^T \Sigma^{-1} K \varphi(s) + \varphi(s)^T K^T \Sigma^{-1} a - 0.5 a^T \Sigma^{-1} a + \text{const} \\ &= -0.5 \varphi(s)^T U K \varphi(s) + \varphi(s)^T U a - 0.5 a^T \Sigma^{-1} a + \text{const}, \end{aligned} \quad (9)$$

where  $K = (k_1, \dots, k_N)$  and  $U = K^T \Sigma^{-1}$ . To compute  $\psi(s, a)$  we note that

$$\phi(s, a) = \nabla_\theta \log \pi_\theta(a|s) = \psi(s, a) - \mathbb{E}_{\pi(\cdot|s)} [\psi(s, \cdot)]$$

To get  $\psi(s, a)$  we note that some parts of Eq. (9) and thus of  $\nabla_\theta \log \pi_\theta(a|s)$  do not depend on  $a$ . We ignore those parts for computing  $\psi(s, a)$  since  $\psi(s, a)\theta$  is the state-action value function and action independent parts of the state-action value function do not influence the optimal action choice. Thus we get

$$\psi(s, a) = \nabla_\theta \left( \varphi(s)^T U a - 0.5 a^T \Sigma^{-1} a \right). \quad (10)$$

We then take the gradient w.r.t. the log-linear parameters  $\theta = (\Sigma^{-1}, U)$  resulting in  $\nabla_{\Sigma^{-1}} \log \hat{\pi}(a|s) = -0.5 a a^T$ ,  $\nabla_U \log \hat{\pi}(a|s) = a \varphi(s)^T$ , and  $\psi(s, a) = [-\text{vec}[0.5 a a^T], \text{vec}[a \varphi(s)^T]]^T$ , where  $\text{vec}[\cdot]$  concatenates matrix columns into a column vector.

Note that the variances and the linear parameters of the mean are contained in the parameter vector  $\theta$  and can be updated by the update rule in Eq. (7) explained above. However, for obtaining the update rules for the non-linear parameters  $\beta$ , we first have to compute the compatible basis, that is,

$$\nabla_\beta \log \pi_{\beta, \theta}(a|s) = \partial \psi_\beta(s, a) \theta / \partial \beta - \mathbb{E}_{\pi(\cdot|s)} [\partial \psi_\beta(s, \cdot) \theta / \partial \beta]. \quad (11)$$

Note that due to the log operator the derivative is linear w.r.t. log-linear parameters  $\theta$ . For the Gaussian distribution in Eq. (8) the gradient of the action dependent parts of the log policy in Eq. (11) become

$$\partial \psi_\beta(s, a) \theta / \partial \beta = \frac{\partial}{\partial \beta} \varphi(s)^T U a.$$

Now, in order to find the update rule for the non-linear parameters we will write the update rule for the policy using Eq. (5), and, using the value function formed by multiplying the compatible basis in Eq. (11) by  $w_\beta$  which is the part of the compatible approximation vector

**Algorithm 1** Compatible Policy Search (COPOS).

---

```

Initialize  $\pi_0$ 
for  $i = 1$  to max episodes do
  Sample  $(s, \mathbf{a}, r)$  tuples using  $\pi_{i-1}$ 
  Estimate advantage function  $A_{\pi_{i-1}}(s, \mathbf{a})$  from samples
  Solve  $\mathbf{w} = \mathbf{F}^{-1} \nabla J_{PG}(\pi_i)$ 
  Use compatible value function to solve Lagrange multipliers  $\eta$  and  $\omega$  for Eq. (4)
  Update  $\pi_i$  using  $\mathbf{w}$ ,  $\eta$  and  $\omega$  based on Eqs. (7) and (16)
end for

```

---

that is responsible for  $\beta$ :

$$\pi_{\beta, \theta}(\mathbf{a}|s) \propto \pi_{\beta_{\text{old}}, \theta_{\text{old}}}(\mathbf{a}|s)^{\eta/(\eta+\omega)} \exp\left(\left(\nabla_{\beta_{\text{old}}} \psi_{\beta_{\text{old}}}(s, \mathbf{a}) \theta_{\text{old}}\right) \mathbf{w}_{\beta}\right. \\ \left. - \mathbb{E}_{\pi(\cdot|s)}\left[\left(\nabla_{\beta_{\text{old}}} \psi_{\beta_{\text{old}}}(s, \mathbf{a}) \theta_{\text{old}}\right)\right] \mathbf{w}_{\beta}\right) \quad (12)$$

$$\propto \exp\left(\frac{\eta \psi_{\beta_{\text{old}}}(s, \mathbf{a}) \theta_{\text{old}}}{\eta + \omega}\right) \exp\left(\frac{\nabla_{\beta_{\text{old}}}(\psi_{\beta_{\text{old}}}(s, \mathbf{a}) \theta_{\text{old}}) \mathbf{w}_{\beta}}{\eta + \omega}\right) \quad (13)$$

$$= \exp\left(\frac{\eta \psi_{\beta_{\text{old}}}(s, \mathbf{a}) \theta_{\text{old}} + \nabla_{\beta_{\text{old}}}(\psi_{\beta_{\text{old}}}(s, \mathbf{a}) \theta_{\text{old}}) \mathbf{w}_{\beta}}{\eta + \omega}\right) \\ = \exp\left(\frac{\eta}{\eta + \omega} \left(\psi_{\beta_{\text{old}}}(s, \mathbf{a}) + \nabla_{\beta_{\text{old}}}(\psi_{\beta_{\text{old}}}(s, \mathbf{a}) \mathbf{w}_{\beta}/\eta)\right) \theta_{\text{old}}\right) \quad (14)$$

$$\approx \exp\left(\frac{\eta}{\eta + \omega} \psi_{\beta_{\text{old}} + \mathbf{w}_{\beta}/\eta}(s, \mathbf{a}) \theta_{\text{old}}\right), \quad (15)$$

where we dropped action independent parts, which can be seen as part of the distribution normalization, from Eq. (12) to Eq. (13). Note that Eq. (15) represents the first order Taylor approximation of Eq. (14) at  $\mathbf{w}_{\beta}/\eta = 0$ . Moreover, note that rescaling of the energy function  $\psi_{\beta}(s, \mathbf{a}) \theta$  is implemented by the update of the parameters  $\theta$  and hence can often be ignored for the update for  $\beta$ . The approximate update rule for  $\beta$  is thus

$$\beta = \beta_{\text{old}} + \mathbf{w}_{\beta}/\eta. \quad (16)$$

Hence, we can conclude that the natural gradient is an approximate trust region solution for the non-linear parameters  $\beta$  as the first order Taylor approximation of the energy function is replaced by the real energy function after the update. Still, for the parameters  $\theta$ , which in the end dominate the mean and covariance of the policy, the natural gradient is the exact trust region solution.

### 3.3 Compatible value function approximation in practice

Algorithm 1 shows the Compatible Policy Search (COPOS) approach (see Appendix B for a more detailed description of the discrete action algorithm version). In COPOS, for the policy updates in Eqs. (7) and (16), we need to find  $\mathbf{w}$ ,  $\eta$ , and  $\omega$ . For estimating  $\mathbf{w}$  we could use the compatible function approximation. In this paper, we do not estimate the value function explicitly but instead estimate  $\mathbf{w}$  as a natural gradient using the conjugate gradient method which removes the need for computing the inverse of the Fisher information matrix explicitly [see for example Schulman et al. 2015]. As discussed before  $\eta$  and  $\omega$  are Lagrange multipliers associated with the KL-divergence and entropy bounds. In the log-linear case with compatible natural parameters, we can compute them exactly using the dual of the optimization objective



Eq. (4) and in the non-linear case approximately. In the continuous action case, the basic dual includes integration over both actions and states but we can integrate over actions in closed form due to the compatible value function: we can eliminate terms which do not depend on the action. The dual resolves into an integral over just states allowing computing  $\eta$  and  $\omega$  efficiently. Please, see “Appendix A” for more details. Since  $\eta$  is an approximation for the non-linear parameters, we performed in the experiments for the continuous action case an additional alternating optimization twice: (1) we did a binary search to satisfy the KL-divergence bound while keeping  $\omega$  fixed, (2) we re-optimized  $\omega$  (exactly, since  $\omega$  depends only on log-linear parameters) keeping  $\eta$  fixed. For discrete actions it was sufficient to perform only an additional line search to update the non-linear parameters.

## 4 Analysis and illustration of the update rules

We will now analyze the performance of both update rules, with and without entropy, in more detail with a simple stateless Gaussian policy  $\pi(a) \propto \exp(-0.5Ba^2 + ba)$  for a scalar action  $a$ . Our real reward function  $R(a) = -0.5Ra^2 + ra$  is also quadratic in the actions. We assume an infinite number of samples to perfectly estimate the compatible function approximation. In this case  $\tilde{G}_w$  is given by  $\tilde{G}_w(a) = R(a) = -0.5Ra^2 + ra$  and  $w = [R, r]$ . The reward function maximum is  $a^* = R^{-1}r$ .

*Natural gradients* For now, we will analyze the performance of the natural gradient if  $\eta$  is a constant and not optimized for a given KL-bound. After  $n$  update steps, the parameters of the policy are given by  $B_n = B_0 + nR/\eta$ ,  $b_n = b_0 + nr/\eta$ . The distance between the mean  $\mu_n = B_n^{-1}b_n$  of the policy and the optimal solution  $a^*$  is

$$d_n = \mu_n - a^* = \frac{b_0R - rB_0}{R(B_0 + nR/\eta)} = O\left(\frac{c_0}{c_1 + c_2n}\right).$$

We can see that the learned solution approaches the optimal solution, however, very slowly and heavily depending on the precision  $B_0$  of the initial solution. The reason for this effect can be explained by the update rule of the precision. As we can see, the precision  $B_n$  is increased at every update step. This shrinking variance in turn decreases the step-size for the next update.

*Entropy regularization* Here we provide the derivation of  $d_n$  for the entropy regularization case. We perform a similar analysis for the entropy regularization update rule. We start with constant parameters  $\eta$  and  $\omega$  and later consider the case with the trust region. The distance  $d_n = \mu_n - a^*$  is again a function of  $n$ . The updates for the entropy regularization result in the following parameters after  $n$  iterations

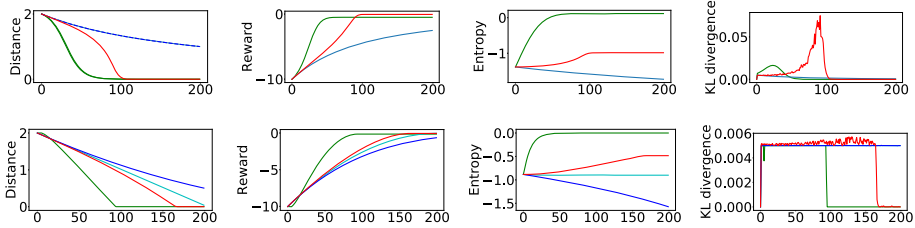
$$B_n = B_0 \frac{\eta^n}{(\eta + \omega)^n} + R/\eta \left(1 - \frac{\eta^n}{(\eta + \omega)^n}\right) / \left(1 - \frac{\eta}{\eta + \omega}\right),$$

$$b_n = b_0 \frac{\eta^n}{(\eta + \omega)^n} + r/\eta \left(1 - \frac{\eta^n}{(\eta + \omega)^n}\right) / \left(1 - \frac{\eta}{\eta + \omega}\right).$$

The distance  $d_n = \mu_n - a^*$  can again be expressed as a function of  $n$ :

$$d_n = \frac{b_0 - rB_0/R}{B_0 + R((\eta + \omega)^n/\eta^n - 1)(\eta + \omega)/(\eta\omega)} = O\left(\frac{c_0}{c_1 + c_2^n}\right), \quad (17)$$

with  $c_2 > 1$ . Hence, also this update rule converges to the correct solution but contrary to the natural gradient, the part of the denominator that depends on  $n$  grows exponentially. As the



**Fig. 1** Comparison of different update rules with and without entropy regularization in the toy example of Sect. 4. The figures show the “Distance” between optimal and current policy mean [see Eq. (17)], the expected “Reward”, the expected “Entropy”, and the expected “KL-divergence” between previous and current policy over 200 iterations (x-axis). Top: Policy updates with constant learning rates and no trust region. Comparison of the natural gradient (blue), natural gradient with entropy regularization (green) and vanilla policy gradient (red). Bottom Policy updates with trust region. Comparison of the natural gradient (blue), natural gradient where the entropy is controlled to a set-value (green), natural gradient with zero entropy loss (cyan) and vanilla policy gradient (red). To summarize, without entropy regularization the natural gradient decreases the entropy too fast (Color figure online)

old parameter vector is always multiplied by a factor smaller than one, the influence of the initial precision matrix  $B_0$  vanishes while  $B_0$  dominates natural gradient convergence. While the natural gradient always decreases variance, entropy regularization avoids the entropy loss and can even increase variance.

**Empirical evaluation of constant updates** We plotted the behavior of the algorithms, and standard policy gradient, for this simple toy task in Fig. 1 (top). We use  $\eta = 10$  and  $\omega = 1$  for the natural gradient and the entropy regularization and a learning rate of  $\alpha = 1000$  for the policy gradient. We estimate the standard policy gradients from 1000 samples. Entropy regularization performs favorably speeding up learning in the beginning by increasing the entropy. With constant parameters  $\eta$  and  $\omega$ , the algorithm drives the entropy to a given target-value. The policy gradient performs better than the natural gradient as it does not reduce the variance all the time and even increases the variance. However, the KL-divergence of the standard policy gradient appears uncontrolled.

**Empirical evaluation of trust region updates** In the trust region case, we minimized the Lagrange dual at each iteration yielding  $\eta$  and  $\omega$ . We chose at each iteration the highest policy gradient learning rate where the KL-bound was still met. For entropy regularization we tested two setups: (1) We fixed the entropy of the policy (that is,  $\gamma = 0$ ), (2) The entropy of the policy was slowly driven to 0. Figure 1 (bottom) shows the results. The natural gradient still suffers from slow convergence due to decreasing the entropy of the policy gradually. The standard gradient again performs better as it increases the entropy outperforming even the zero entropy loss natural gradient. For entropy control, even more sophisticated scheduling could be used such as the step-size control of CMA-ES (Hansen and Ostermeier 2001) as a heuristic that works well.

## 5 Related work

Similar to classical reinforcement learning the leading contenders in deep reinforcement learning can be divided into value based-function methods such as Q-learning with deep Q-Network (DQN) (Mnih et al. 2015), actor-critic methods (Wu et al. 2017; Tangkaratt et al. 2018; Abdolmaleki et al. 2018), policy gradient methods such as deep deterministic policy gradient (DDPG) (Silver et al. 2014; Lillicrap et al. 2015) and policy search methods based

on information theoretic/trust region methods, such as proximal policy optimization (PPO) (Schulman et al. 2017) and trust region policy optimization (TRPO) (Schulman et al. 2015).

Trust region optimization was introduced in the relative entropy policy search (REPS) method (Peters et al. 2010). TRPO and TNPG (Schulman et al. 2015) are the first methods to apply trust region optimization successfully to neural networks. In contrast to TRPO and TNPG, we derive our method from the compatible value function approximation perspective. TRPO and TNPG differ from our approach, in that they do not use an entropy constraint and do not consider the difference between the log-linear and non-linear parameters for their update. On the technical level, compared to TRPO, we can update the log-linear parameters (output layer of neural network and the covariance) with an exact update step while TRPO does a line search to find the update step. Moreover, for the covariance we can find an exact update to enforce a specific entropy and thus control exploration while TRPO does not bound the entropy, only the KL-divergence. PPO also applies an adaptive KL penalty term.

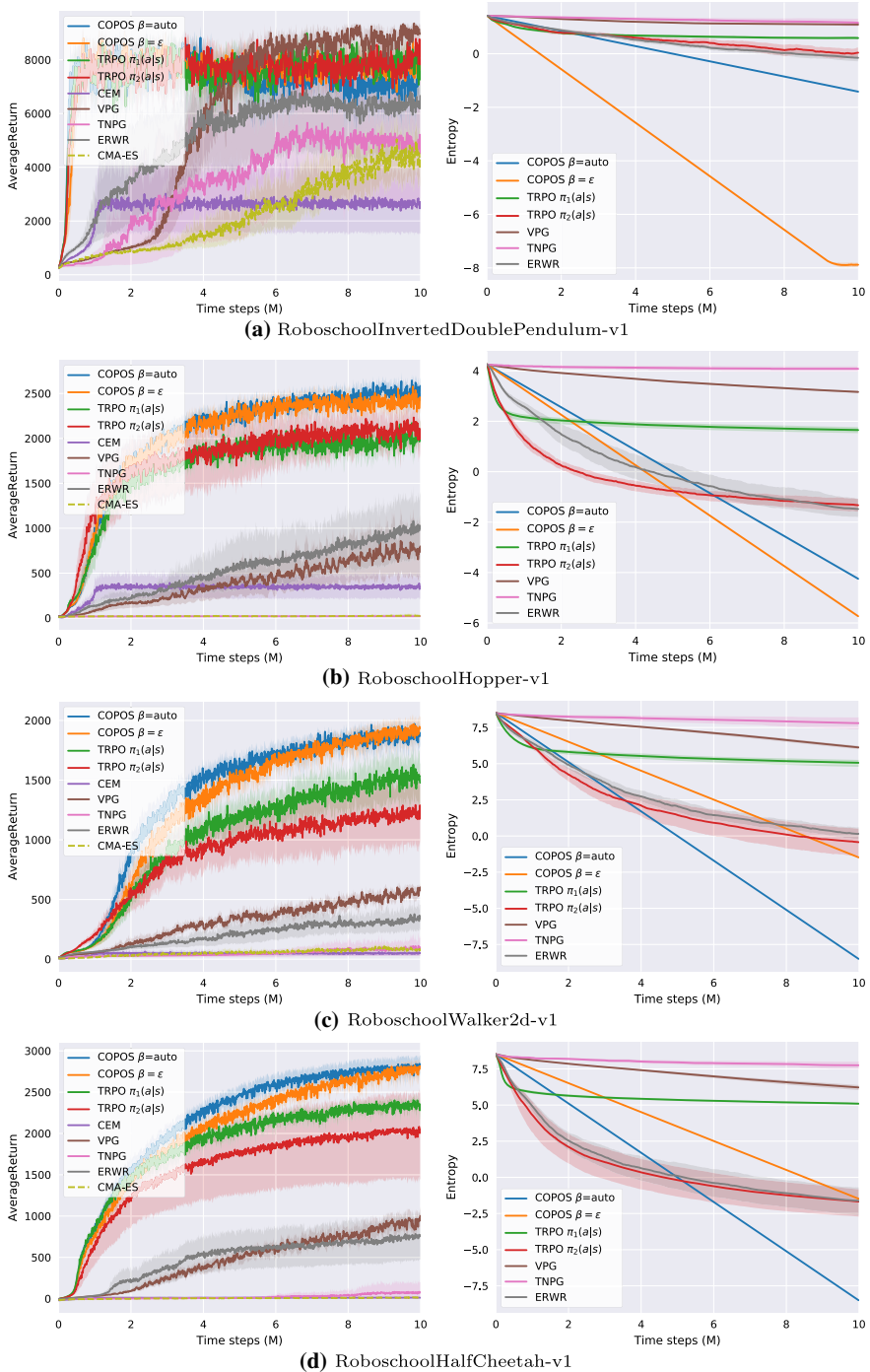
Kakade (2001); Bagnell and Schneider (2003); Peters and Schaal (2008); Geist and Pietquin (2010) have also suggested similar update rules based on the natural gradient for the policy gradient framework. Wu et al. (2017) applied approximate natural gradient updates to both the actor and critic in an actor-critic framework but did not utilize compatible value functions or an entropy bound. Peters and Schaal (2008); Geist and Pietquin (2010) investigated the idea of compatible value functions in combination with the natural gradient but used manual learning rates instead of trust region optimization. The approaches in (Abdolmaleki et al. 2015; Akroun et al. 2016) use an entropy bound similar to ours. However, the approach in (Abdolmaleki et al. 2015) is a stochastic search method, that is, it ignores sequential decisions and views the problem as black-box optimization, and the approach in (Akroun et al. 2016) is restricted to trajectory optimization. Moreover, both of these approaches do not explicitly handle non-linear parameters such as those found in neural networks. The entropy bound used in (Tangkaratt et al. 2018) is similar to ours, however, their method depends on second order approximations of a deep Q-function, resulting in a much more complex policy update that can suffer from the instabilities of learning a non-linear Q-function.

For exploration one can in general add an entropy term to the objective. In the experiments, we compare against TRPO with this additive entropy term. In preliminary experiments, to control entropy in TRPO, we also combined the entropy and KL-divergence constraints into a single constraint without success.

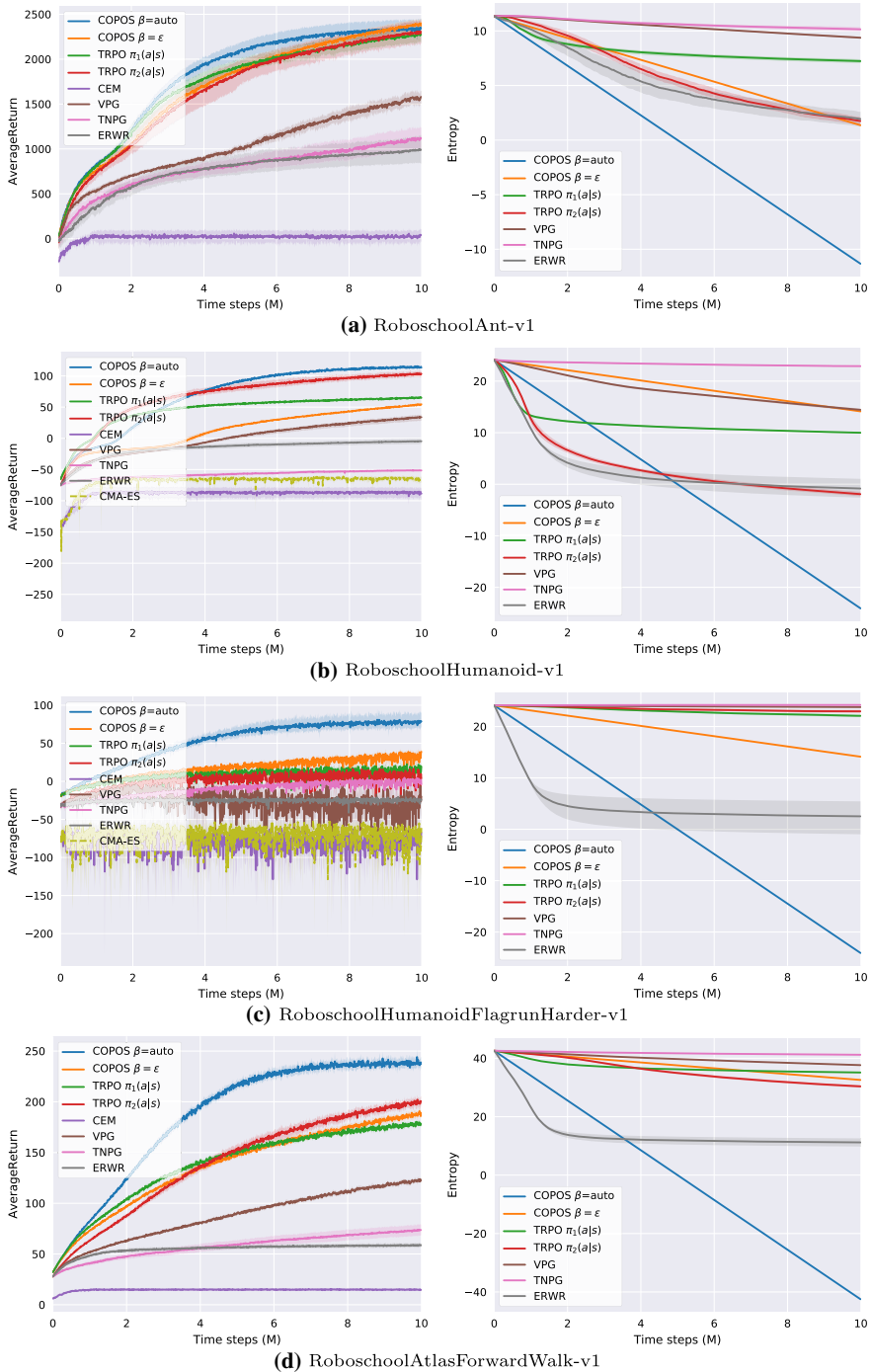
## 6 Experiments

In the experiments, we focused on investigating the following research question: Does the proposed entropy regularization approach help to improve performance compared to other methods which do not control the entropy explicitly? For selecting comparison methods we followed (Duan et al. 2016) and took four gradient based methods: Trust Region Policy Optimization (TRPO) (Schulman et al. 2015), Truncated Natural Policy Gradient (Duan et al. 2016; Schulman et al. 2015), REINFORCE (VPG) (Williams 1992), Reward-Weighted Regression (RWR) (Kober and Peters 2009) and two gradient-free black box optimization methods: Cross Entropy Method (CEM) (Rubinstein 1999), Covariance Matrix Adaption Evolution Strategy (CMA-ES) (Hansen and Ostermeier 2001). We used rllab<sup>1</sup> for algorithm implementation. We ran experiments in both challenging continuous control tasks and discrete partially observable tasks which we discuss next.

<sup>1</sup> <http://rllab.readthedocs.io/en/latest/>.



**Fig. 2** Average return and differential entropy over 10 random seeds of comparison methods in continuous Roboschool tasks (see, Fig. 3 for the other continuous tasks and Table 1 for a summary). Shaded area denotes the bootstrapped 95% CI. Algorithms were executed for 1000 iterations with 10,000 time steps (samples) in each iteration



**Fig. 3** Average return and differential entropy of comparison methods over 10 random seeds in continuous Roboschool tasks (see, Fig. 2 for the other continuous tasks and Table 1 for a summary). Shaded area denotes the bootstrapped 95% CI. Algorithms were executed for 1000 iterations with 10, 000 time steps (samples) in each iteration

**Table 1** Continuous control environments

	COPOS $\beta = \text{auto}$	COPOS $\beta = \epsilon$	TRPO $\pi_1(a s)$
RoboInvertedDoublePendulum-v1	7107 $\pm$ 416	7722 $\pm$ 235.3	7904 $\pm$ 192.4
RoboHopper-v1	<b>2514 <math>\pm</math> 35.11</b>	<b>2427 <math>\pm</math> 47.96</b>	2005 $\pm$ 26.04
RoboWalker2d-v1	<b>1878 <math>\pm</math> 34.59</b>	<b>1906 <math>\pm</math> 41.48</b>	1518 $\pm$ 93.11
RoboHalfCheetah-v1	<b>2804 <math>\pm</math> 57.27</b>	<b>2772 <math>\pm</math> 42.23</b>	2341 $\pm$ 16.52
RoboAnt-v1	<b>2335 <math>\pm</math> 49.69</b>	<b>2375 <math>\pm</math> 26.88</b>	<b>2263 <math>\pm</math> 48.09</b>
RoboHumanoid-v1	<b>113.85 <math>\pm</math> 0.94</b>	52.92 $\pm$ 0.31	64.77 $\pm$ 0.38
RoboHumanoidFlagrunHarder-v1	<b>77.72 <math>\pm</math> 4.86</b>	32.14 $\pm$ 1.34	14.51 $\pm$ 1.25
RoboAtlasForwardWalk-v1	<b>238.1 <math>\pm</math> 2.08</b>	186.8 $\pm$ 1.18	177.7 $\pm$ 0.44
	TRPO $\pi_2(a s)$	CMA-ES	VPG
RoboInvertedDoublePendulum-v1	8052 $\pm$ 172.3	4382 $\pm$ 379.2	<b>9020 <math>\pm</math> 35.10</b>
RoboHopper-v1	2106 $\pm$ 85.59	25.87 $\pm$ 8.66	751.4 $\pm$ 145.4
RoboWalker2d-v1	1223 $\pm$ 139.0	82.38 $\pm$ 9.72	559.7 $\pm$ 13.55
RoboHalfCheetah-v1	2024 $\pm$ 261.2	13.38 $\pm$ 3.88	918.1 $\pm$ 43.57
RoboAnt-v1	<b>2291 <math>\pm</math> 65.66</b>	Out of memory	1558 $\pm$ 35.71
RoboHumanoid-v1	102.5 $\pm$ 2.16	- 65.07 $\pm$ 1.71	32.72 $\pm$ 2.47
RoboHumanoidFlagrunHarder-v1	5.04 $\pm$ 2.70	- 74.62 $\pm$ 3.28	- 31.62 $\pm$ 4.59
RoboAtlasForwardWalk-v1	198.4 $\pm$ 1.94	Out of memory	121.5 $\pm$ 0.92
	CEM	TNPG	ERWR
RoboInvertedDoublePendulum-v1	2643 $\pm$ 628.9	4866 $\pm$ 1178	6367 $\pm$ 937.4
RoboHopper-v1	346.6 $\pm$ 66.89	20.55 $\pm$ 2.42	985.1 $\pm$ 173.1
RoboWalker2d-v1	49.03 $\pm$ 10.69	90.36 $\pm$ 35.52	333.3 $\pm$ 47.73
RoboHalfCheetah-v1	11.27 $\pm$ 2.93	72.22 $\pm$ 56.48	747.3 $\pm$ 137.8
RoboAnt-v1	25.17 $\pm$ 35.09	1102 $\pm$ 64.61	982.6 $\pm$ 74.23
RoboHumanoid-v1	- 87.19 $\pm$ 4.21	- 51.39 $\pm$ 1.01	- 4.95 $\pm$ 2.16
RoboHumanoidFlagrunHarder-v1	- 78.00 $\pm$ 4.74	- 0.25 $\pm$ 0.68	- 24.59 $\pm$ 0.65
RoboAtlasForwardWalk-v1	14.90 $\pm$ 0.60	72.96 $\pm$ 2.75	58.54 $\pm$ 1.07

Mean of the average return over 50 last iterations  $\pm$  standard error over 10 random seeds. Bold denotes: no statistically significant difference to the best result (Welch's t-test with  $p < 0.05$ )

*Continuous tasks* In the continuous case, we ran experiments in eight different Roboschool<sup>2</sup> environments which provide continuous control tasks of increasing difficulty and action dimensions without requiring a paid license.

We ran all evaluations, 10 random seeds for each method, for 1000 iterations of 10,000 samples each. In all problems, we used the Gaussian policy defined in Eq. (8) for COPOS and TRPO (denoted by  $\pi_1(a|s)$ ) with max(10, action dimensions) neural network outputs as basis functions, a neural network with two hidden layers each containing 32 tanh-neurons, and a diagonal precision matrix. For TRPO and other methods, except COPOS, we also evaluated a policy, denoted for TRPO by  $\pi_2(a|s)$ , where the neural network directly specifies the mean and the diagonal covariance is parameterized with log standard deviations. In the experiments, we used high identical initial variances (we tried others without success) for all

<sup>2</sup> <https://github.com/openai/roboschool>.

**Table 2** Average discounted return on discrete control FVRS instances (fully and partially observable)

	COPOS	TRPO	TRPO ent reg	TNPG
5 × 5 full	<b>2.14 ± 0.08</b>	1.50 ± 0.23	1.47 ± 0.22	1.43 ± 0.28
5 × 5 noise	<b>1.94 ± 0.12</b>	1.24 ± 0.02	1.24 ± 0.01	1.24 ± 0.01
5 × 7 full	<b>2.66 ± 0.14</b>	1.80 ± 0.20	1.92 ± 0.17	1.87 ± 0.16
5 × 7 noise	<b>2.45 ± 0.10</b>	2.01 ± 0.02	2.02 ± 0.02	2.01 ± 0.03
7 × 8 full	<b>1.66 ± 0.17</b>	1.28 ± 0.32	1.31 ± 0.20	1.19 ± 0.26
7 × 8 noise	<b>1.32 ± 0.15</b>	<b>1.22 ± 0.28</b>	<b>1.36 ± 0.16</b>	<b>1.18 ± 0.22</b>

Bold denotes: no statistically significant difference to the best result (Welch's t-test with  $p < 0.05$ )

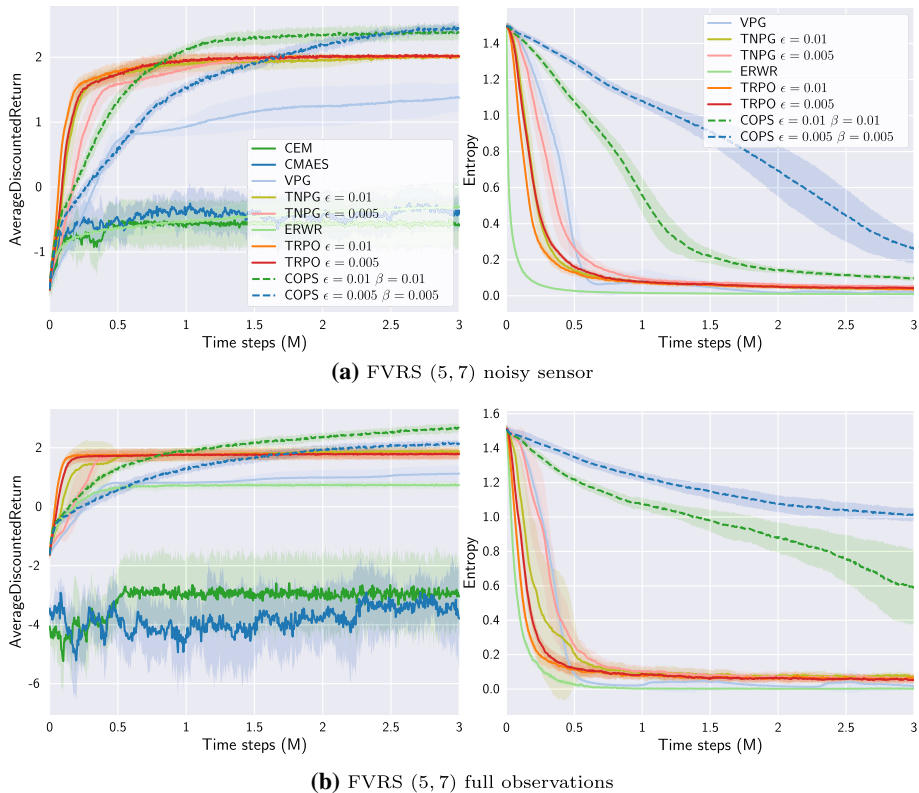
policies. We set  $\epsilon = 0.01$  (Schulman et al. 2015) in all experiments. For COPOS we used two equality entropy constraints:  $\beta = \epsilon$  and  $\beta = \text{auto}$ . In  $\beta = \text{auto}$ , we assume positive initial entropy and schedule the entropy to be the negative initial entropy after 1000 iterations. Since we always initialize the variances to one, higher dimensional problems have higher initial entropy. Thus  $\beta = \text{auto}$  reduces the entropy faster for high dimensional problems effectively scaling the reduction with dimensionality. Table 1 summarizes the results in continuous tasks: COPOS outperforms comparison methods in most of the environments. Figures 2 and 3 show learning curves and performance of COPOS compared to the other methods. COPOS prevents both too fast, and, too slow entropy reduction while outperforming comparison methods. Table 4 in “Appendix C” shows additional results for experiments where different constant entropy bonuses were added to the reward function of TRPO without success highlighting the necessity of principled entropy control.

*Discrete control task* Partial observability often requires efficient exploration due to non-myopic actions yielding long term rewards which is challenging for model-free methods. The Field Vision Rock Sample (FVRS) (Ross et al. 2008) task is a partially observable Markov decision process (POMDP) benchmark task. For the discrete action experiments we used as policy a softmax policy with a fully connected feed forward neural network consisting of 2 hidden layers with 30 tanh nonlinearities each. The input to the neural network is the observation history and the current position of the agent in the grid. To obtain the hyperparameters  $\beta$ ,  $\epsilon$ , and the scaling factor for TRPO with additive entropy regularization, denoted with “TRPO ent reg”, we performed a grid search on smaller instances of FVRS. See “Appendix B” for more details about the setup. Results in Table 2 and Fig. 4 show that COPOS outperforms the comparison methods due to maintaining higher entropy. FVRS has been used with model-based online POMDP algorithms (Ross et al. 2008) but not with model-free algorithms. The best model-based results in (Ross et al. 2008) (scaled to correspond to our rewards, COPOS in parentheses) are 2.275 (1.94) in FVRS(5,5) and 2.34 (2.45) in FVRS(5,7).

## 7 Conclusions and future work

We showed that when we use the natural parameterization of a standard exponential policy distribution in combination with compatible value function approximation, the natural gradient and trust region optimization are equivalent. Furthermore, we demonstrated that natural gradient updates may reduce the entropy of the policy according to a schedule which can lead to premature convergence. To combat the problem of bad entropy scheduling in trust region





**Fig. 4** Average discounted return and Shannon entropy for both FVRS  $5 \times 7$  with a noisy sensor and full observations over 10 random seeds. Shaded area denotes the bootstrapped 95% CI. Algorithms were executed for 600 iterations with 5000 time steps (samples) in each iteration

methods we proposed a new compatible policy search method called COPOS that can control the entropy of the policy using an entropy bound. In both challenging high dimensional continuous and discrete tasks the approach yielded state-of-the-art results due to better entropy control. In future work, an exciting direction is to apply efficient approximations to compute the natural gradient (Bernacchi et al. 2018). Moreover, we have started work on applying the proposed algorithm in challenging partially observable environments found for example in autonomous driving where exploration and sample efficiency is crucial for finding high quality policies (Dosovitskiy et al. 2017).

**Acknowledgements** This work was supported by EU Horizon 2020 project RoMaNS and ERC StG SKILLS4ROBOTS, project references #645582 and #640554, and, by German Research Foundation project PA 3179/1-1 (ROBOLEAP).

**Open Access** This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.



## A Solution for the Lagrange multipliers

In order to compute a solution to the optimization objective with a KL-divergence and an entropy bound, we solve, using the dual of the problem, for the Lagrange multipliers associated with the bounds. We first discuss for the continuous action case how we optimize the multipliers exactly in the case of only log-linear parameters, continue with how we find an approximate solution in the case of also non-linear parameters, and then discuss the discrete action case.

### A.1 Computing Lagrange multipliers $\eta$ and $\omega$ for log-linear parameters

Minimize the dual of the optimization objective (see e.g. Akrou et al. (2016) for a similar dual)

$$g_t(\eta, \omega) = \eta\epsilon - \omega\beta + (\eta + \omega) \times \int \tilde{p}_t(s) \log \left( \int \pi(\mathbf{a}|s)^{\eta/(\eta+\omega)} \exp \left( \tilde{Q}_t(s, \mathbf{a})/(\eta + \omega) \right) d\mathbf{a} \right) ds \quad (18)$$

w.r.t.  $\eta$  and  $\omega$ . Note that action independent parts of  $\log \pi(\mathbf{a}|s)$  in Eq. (18) do not have an effect on the choice of  $\eta$  and  $\omega$  and we will discard them.

For a Gaussian policy

$$\pi(\mathbf{a}|s) = \mathcal{N} \left( \mathbf{a} \mid \boldsymbol{\mu} = \mathbf{K}\boldsymbol{\varphi}(s) = \sum_i \varphi_i(s) \mathbf{k}_i, \boldsymbol{\Sigma} \right) \quad (19)$$

we get

$$\begin{aligned} \tilde{Q}_t(s, \mathbf{a}) &= \boldsymbol{\psi}(s, \mathbf{a})^T \mathbf{w} = \begin{bmatrix} -\text{vec}[0.5\mathbf{a}\mathbf{a}^T] \\ \text{vec}[\mathbf{a}\boldsymbol{\varphi}(s)^T] \end{bmatrix}^T \begin{bmatrix} \mathbf{w}_1 \\ \mathbf{w}_2 \end{bmatrix} \\ &= -0.5\mathbf{a}^T \mathbf{W}_{aa} \mathbf{a} + \boldsymbol{\varphi}(s)^T \mathbf{W}_{sa} \mathbf{a} \end{aligned} \quad (20)$$

$$\begin{aligned} g_t(\eta, \omega) &= \eta\epsilon - \omega\beta + (\eta + \omega) \\ &\times \int \tilde{p}_t(s) \log \left( \int C_{\pi}^{\frac{\eta}{\eta+\omega}} \exp \left( -0.5\eta/(\eta + \omega) (\mathbf{a} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{a} - \boldsymbol{\mu}) \right) \right. \\ &\left. \exp \left( -0.5\mathbf{a}^T \mathbf{W}_{aa}/(\eta + \omega) \mathbf{a} + \boldsymbol{\varphi}(s)^T \mathbf{W}_{sa}/(\eta + \omega) \mathbf{a} \right) d\mathbf{a} \right) ds \end{aligned} \quad (21)$$

$$\begin{aligned} g_t(\eta, \omega) &= \eta\epsilon - \omega\beta + (\eta + \omega) \int \tilde{p}_t(s) \log \left( \int C_{\pi}^{\frac{\eta}{\eta+\omega}} \right. \\ &\exp \left( \frac{\eta}{\eta + \omega} (-0.5\mathbf{a}^T \boldsymbol{\Sigma}^{-1} \mathbf{a} + \boldsymbol{\mu}^T \boldsymbol{\Sigma}^{-1} \mathbf{a} - 0.5\boldsymbol{\mu}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}) \right) \\ &\left. \exp \left( -0.5\mathbf{a}^T \mathbf{W}_{aa}/(\eta + \omega) \mathbf{a} + \boldsymbol{\varphi}(s)^T \mathbf{W}_{sa}/(\eta + \omega) \mathbf{a} \right) d\mathbf{a} \right) ds \end{aligned} \quad (22)$$

$$\begin{aligned} g_t(\eta, \omega) &= \eta\epsilon - \omega\beta + (\eta + \omega) \\ &\times \int \tilde{p}_t(s) \log \left( \int C_{\pi}^{\frac{\eta}{\eta+\omega}} \exp \left( \frac{1}{\eta + \omega} \left( -0.5\mathbf{a}^T (\eta \boldsymbol{\Sigma}^{-1} + \mathbf{W}_{aa}) \mathbf{a} \right) \right) \right. \end{aligned}$$

$$+ \left( \eta \boldsymbol{\varphi}(s)^T \mathbf{K}^T \boldsymbol{\Sigma}^{-1} + \boldsymbol{\varphi}(s)^T \mathbf{W}_{sa} \right) \mathbf{a} - 0.5 \eta \boldsymbol{\varphi}(s)^T \mathbf{K}^T \boldsymbol{\Sigma}^{-1} \mathbf{K} \boldsymbol{\varphi}(s) \Big) d\mathbf{a} \Big) ds \quad (23)$$

$$g_t(\eta, \omega) = \eta\epsilon - \omega\beta + (\eta + \omega) \int \tilde{p}_t(s) \log \left( \int C_{\pi}^{\frac{\eta}{\eta+\omega}} \exp \left( \frac{1}{\eta + \omega} \left( -0.5 \mathbf{a}^T \mathbf{H}_{aa} \mathbf{a} + \boldsymbol{\varphi}(s)^T \mathbf{H}_{sa} \mathbf{a} - 0.5 \boldsymbol{\varphi}(s)^T \mathbf{H}_{ss} \boldsymbol{\varphi}(s) \right) \right) d\mathbf{a} \right) ds \quad (24)$$

$$g_t(\eta, \omega) = \eta\epsilon - \omega\beta + (\eta + \omega) \int \tilde{p}_t(s) \log \left( \int C_{\pi}^{\frac{\eta}{\eta+\omega}} \exp \left( \frac{1}{\eta + \omega} \left( -0.5 (\mathbf{a} - \mathbf{H}_{aa}^{-1} \mathbf{H}_{sa}^T \boldsymbol{\varphi}(s))^T \mathbf{H}_{aa} (\mathbf{a} - \mathbf{H}_{aa}^{-1} \mathbf{H}_{sa}^T \boldsymbol{\varphi}(s)) \right. \right. \right. \\ \left. \left. \left. + 0.5 \boldsymbol{\varphi}(s)^T \mathbf{H}_{sa} \mathbf{H}_{aa}^{-1} \mathbf{H}_{sa}^T \boldsymbol{\varphi}(s) - 0.5 \boldsymbol{\varphi}(s)^T \mathbf{H}_{ss} \boldsymbol{\varphi}(s) \right) \right) d\mathbf{a} \right) ds \quad (25)$$

$$g_t(\eta, \omega) = \eta\epsilon - \omega\beta + (\eta + \omega) \times \int \tilde{p}_t(s) \log \left( \exp \left( \frac{1}{\eta + \omega} \left( 0.5 \boldsymbol{\varphi}(s)^T (\mathbf{H}_{sa} \mathbf{H}_{aa}^{-1} \mathbf{H}_{sa}^T - \mathbf{H}_{ss}) \boldsymbol{\varphi}(s) \right) \right) \right. \\ \left. ((2\pi)^{-k/2} |\boldsymbol{\Sigma}|^{-0.5})^{\frac{\eta}{\eta+\omega}} / ((2\pi)^{-k/2} |(\eta + \omega) \mathbf{H}_{aa}^{-1}|^{-0.5}) \right) ds \quad (26)$$

$$g_t(\eta, \omega) = \eta\epsilon - \omega\beta + (\eta + \omega) \int \tilde{p}_t(s) \left( \frac{1}{\eta + \omega} 0.5 \boldsymbol{\varphi}(s)^T (\mathbf{H}_{sa} \mathbf{H}_{aa}^{-1} \mathbf{H}_{sa}^T - \mathbf{H}_{ss}) \boldsymbol{\varphi}(s) \right. \\ \left. + \frac{\eta}{\eta + \omega} \log((2\pi)^{-k/2} |\boldsymbol{\Sigma}|^{-0.5}) - \log((2\pi)^{-k/2} |(\eta + \omega) \mathbf{H}_{aa}^{-1}|^{-0.5}) \right) ds, \quad (27)$$

$$g_t(\eta, \omega) = \eta\epsilon - \omega\beta + \int \tilde{p}_t(s) \left( 0.5 \boldsymbol{\varphi}(s)^T (\mathbf{H}_{sa} \mathbf{H}_{aa}^{-1} \mathbf{H}_{sa}^T - \mathbf{H}_{ss}) \boldsymbol{\varphi}(s) \right) ds \\ + \eta \log((2\pi)^{-k/2} |\boldsymbol{\Sigma}|^{-0.5}) - (\eta + \omega) \log((2\pi)^{-k/2} |(\eta + \omega) \mathbf{H}_{aa}^{-1}|^{-0.5}), \quad (28)$$

$$g_t(\eta, \omega) = \eta\epsilon - \omega\beta + \int \tilde{p}_t(s) \left( 0.5 \boldsymbol{\varphi}(s)^T (\mathbf{H}_{sa} \mathbf{H}_{aa}^{-1} \mathbf{H}_{sa}^T - \mathbf{H}_{ss}) \boldsymbol{\varphi}(s) \right) ds \\ - 0.5 \eta \log |2\pi \boldsymbol{\Sigma}| + 0.5 (\eta + \omega) \log |2\pi (\eta + \omega) \mathbf{H}_{aa}^{-1}|, \quad (29)$$

where

$$\mathbf{H}_{aa} = \eta \boldsymbol{\Sigma}^{-1} + \mathbf{W}_{aa} \quad (30)$$

$$\mathbf{H}_{sa} = \eta \mathbf{K}^T \boldsymbol{\Sigma}^{-1} + \mathbf{W}_{sa} \quad (31)$$

$$\mathbf{H}_{ss} = \eta \mathbf{K}^T \boldsymbol{\Sigma}^{-1} \mathbf{K} \quad (32)$$

and  $k$  is the dimensionality of actions. We got the end result by completing the square.

## A.2 Computing $\eta$ and $\omega$ for non-linear parameters

Similarly to the log-linear parameters we minimize the dual

$$g_t(\eta, \omega) = \eta\epsilon - \omega\beta + (\eta + \omega)$$

$$\times \int \tilde{p}_t(s) \log \left( \int \pi(\mathbf{a}|s)^{\eta/(\eta+\omega)} \exp \left( \tilde{Q}_t(s, \mathbf{a})/(\eta + \omega) \right) d\mathbf{a} \right) ds \quad (33)$$

w.r.t.  $\eta$  and  $\omega$ . As before action independent parts of  $\log \pi(\mathbf{a}|s)$  in Eq. (18) do not have an effect on the choice of  $\eta$  and  $\omega$  and we will discard them.

In our *Linear Gaussian policy with constant covariance*

$$\pi(\mathbf{a}|s) = \mathcal{N} \left( \mathbf{a} \middle| \boldsymbol{\mu} = \mathbf{K}\boldsymbol{\varphi}(s) = \sum_i \varphi_i(s) \mathbf{k}_i, \boldsymbol{\Sigma} \right), \quad (34)$$

we have

$$\log \pi(\mathbf{a}|s) = -0.5\boldsymbol{\varphi}(s)^T \mathbf{U} \mathbf{K} \boldsymbol{\varphi}(s) + \boldsymbol{\varphi}(s)^T \mathbf{U} \mathbf{a} - 0.5\mathbf{a}^T \boldsymbol{\Sigma}^{-1} \mathbf{a} + \text{const}, \quad (35)$$

where  $\text{const} = -\sqrt{(2\pi)^k |\boldsymbol{\Sigma}|}$  and  $\mathbf{U} = \mathbf{K}^T \boldsymbol{\Sigma}^{-1}$ . Therefore,

$$\nabla_{\boldsymbol{\beta}} \log \pi_{\boldsymbol{\beta}, \theta}(\mathbf{a}|s) = \frac{\partial}{\partial \boldsymbol{\beta}} \boldsymbol{\varphi}(s)^T \mathbf{U} \mathbf{a} - \frac{\partial}{\partial \boldsymbol{\beta}} 0.5\boldsymbol{\varphi}(s)^T \mathbf{U} \mathbf{K} \boldsymbol{\varphi}(s), \quad (36)$$

where we are able to split the equation into action-value and value parts depending on whether they depend on  $\mathbf{a}$ . Using the action-value part  $\frac{\partial}{\partial \boldsymbol{\beta}} \boldsymbol{\varphi}(s)^T \mathbf{U} \mathbf{a}$  to estimate  $\mathbf{w}_3$  we get

$$\begin{aligned} \tilde{Q}_t(s, \mathbf{a}) &= \boldsymbol{\psi}(s, \mathbf{a})^T \mathbf{w} = \begin{bmatrix} -\text{vec}[0.5\mathbf{a}\mathbf{a}^T] \\ \text{vec}[\mathbf{a}\boldsymbol{\varphi}(s)^T] \\ \frac{\partial}{\partial \boldsymbol{\beta}} \boldsymbol{\varphi}(s)^T \mathbf{U} \mathbf{a} \end{bmatrix}^T \begin{bmatrix} \mathbf{w}_1 \\ \mathbf{w}_2 \\ \mathbf{w}_3 \end{bmatrix} \\ &= -0.5\mathbf{a}^T \mathbf{W}_{aa} \mathbf{a} + \boldsymbol{\varphi}(s)^T \mathbf{W}_{sa} \mathbf{a} + \mathbf{w}_a(s)^T \mathbf{a}, \end{aligned} \quad (37)$$

where  $\mathbf{w}_a(s)^T = \mathbf{w}_3^T \frac{\partial \boldsymbol{\varphi}(s)^T}{\partial \boldsymbol{\beta}} \mathbf{U}$ . By completing the square we get

$$\begin{aligned} g_t(\eta, \omega) &= \eta\epsilon - \omega\beta + (\eta + \omega) \\ &\times \int \tilde{p}_t(s) \log \left( \int C_{\pi}^{\frac{\eta}{\eta+\omega}} \exp \left( -0.5\eta/(\eta + \omega)(\mathbf{a} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{a} - \boldsymbol{\mu}) \right) \right. \\ &\left. \exp \left( -0.5\mathbf{a}^T \mathbf{W}_{aa}/(\eta + \omega)\mathbf{a} + (\boldsymbol{\varphi}(s)^T \mathbf{W}_{sa} + \mathbf{w}_a(s)^T)/(\eta + \omega)\mathbf{a} \right) d\mathbf{a} \right) ds \end{aligned} \quad (38)$$

$$\begin{aligned} g_t(\eta, \omega) &= \eta\epsilon - \omega\beta + (\eta + \omega) \\ &\times \int \tilde{p}_t(s) \log \left( \int C_{\pi}^{\frac{\eta}{\eta+\omega}} \exp \left( \frac{1}{\eta + \omega} \left( -0.5\mathbf{a}^T (\eta \boldsymbol{\Sigma}^{-1} + \mathbf{W}_{aa}) \mathbf{a} \right. \right. \right. \\ &\left. \left. \left. + (\eta \boldsymbol{\varphi}(s)^T \mathbf{K}^T \boldsymbol{\Sigma}^{-1} + (\boldsymbol{\varphi}(s)^T \mathbf{W}_{sa} + \mathbf{w}_a(s)^T)) \mathbf{a} \right. \right. \right. \\ &\left. \left. \left. - 0.5\eta \boldsymbol{\varphi}(s)^T \mathbf{K}^T \boldsymbol{\Sigma}^{-1} \mathbf{K} \boldsymbol{\varphi}(s) \right) \right) d\mathbf{a} \right) ds \end{aligned} \quad (39)$$

$$\begin{aligned} g_t(\eta, \omega) &= \eta\epsilon - \omega\beta + \int \tilde{p}_t(s) \left( 0.5(\mathbf{h}_a(s)^T \mathbf{H}_{aa}^{-1} \mathbf{h}_a(s) - h_{ss}(s)) \right) ds \\ &- 0.5\eta \log |2\pi \boldsymbol{\Sigma}| + 0.5(\eta + \omega) \log |2\pi(\eta + \omega) \mathbf{H}_{aa}^{-1}|, \end{aligned} \quad (40)$$

where

$$\mathbf{H}_{aa} = \eta \boldsymbol{\Sigma}^{-1} + \mathbf{W}_{aa} \quad (41)$$

$$\mathbf{h}_a(s) = \eta \boldsymbol{\varphi}(s)^T \mathbf{K}^T \boldsymbol{\Sigma}^{-1} + \boldsymbol{\varphi}(s)^T \mathbf{W}_{sa} + \mathbf{w}_a(s)^T \quad (42)$$

$$h_{ss}(s) = \eta \varphi(s)^T \mathbf{K}^T \Sigma^{-1} \mathbf{K} \varphi(s) \quad (43)$$

and  $k$  is the dimensionality of actions.

### A.3 Derivation of the dual for the discrete action case

To derive the dual of our trust region optimization problem with entropy regularization and discrete actions we start with following program, where we replaced the expectation with integrals and use the compatible value function for the returns.

$$\begin{aligned} & \operatorname{argmax}_{\pi_\theta} \int p(s) \int \pi_\theta(\mathbf{a}|s) \tilde{G}_{\mathbf{w}}^{\pi_{\text{old}}}(s, \mathbf{a}) d\mathbf{a} ds \\ & \text{subject to } \int p(s) \int KL(\pi_\theta(\mathbf{a}|s) || \pi_{\theta_{\text{old}}}(\mathbf{a}|s)) d\mathbf{a} ds < \epsilon \\ & \int p(s) \int H(\pi_\theta(\mathbf{a}|s)) - H(\pi_{\theta_{\text{old}}}(\mathbf{a}|s)) d\mathbf{a} ds < \beta \\ & 1 = \int \int p(s) \pi_\theta(\mathbf{a}|s) d\mathbf{a} ds \end{aligned} \quad (44)$$

Since we are in the discrete action case we will be using sums for the brevity of the derivation, but the same derivation can be also done with integrals. Using the method of Lagrange multipliers Boyd and Vandenberghe (2004), we obtain following Lagrange

$$\begin{aligned} L(\pi(\mathbf{a}|s), \eta, \omega, \lambda) = & - \sum_{s, \mathbf{a}} p(s) \pi(\mathbf{a}|s) \tilde{G}_{\mathbf{w}}^{\pi_{\text{old}}}(\mathbf{a}, s) \\ & + \eta \left[ \sum_s p(s) \sum_{\mathbf{a}} \pi(\mathbf{a}|s) \log \left( \frac{\pi(\mathbf{a}|s)}{\pi_{\text{old}}(\mathbf{a}|s)} \right) - \epsilon \right] \\ & + \omega \left[ \sum_{s, \mathbf{a}} p(s) H(\pi_{\text{old}}(\mathbf{a}|s)) + \sum_{s, \mathbf{a}} p(s) \pi(\mathbf{a}|s) \log(\pi(\mathbf{a}|s)) - \beta \right] \\ & + \lambda \left[ \sum_{s, \mathbf{a}} p(s) \pi(\mathbf{a}|s) - 1 \right] \end{aligned} \quad (45)$$

We differentiate now the Lagrange with respect to  $\pi(\mathbf{a}|s)$  and obtain following system

$$\delta_{\pi(\mathbf{a}|s)} L = p(s) \left[ -\tilde{G}_{\mathbf{w}}^{\pi_{\text{old}}}(s, \mathbf{a}) + \eta \log \left( \frac{\pi(\mathbf{a}|s)}{\pi_{\text{old}}(\mathbf{a}|s)} \right) + \eta + \omega \log(\pi(\mathbf{a}|s)) + \omega + \lambda \right]. \quad (46)$$

Setting it to zero and rearranging terms results in

$$\pi(\mathbf{a}|s) = \pi_{\text{old}}(\mathbf{a}|s)^{\frac{\eta}{\eta+\omega}} \exp \left( \frac{\tilde{G}_{\mathbf{w}}^{\pi_{\text{old}}}(s, \mathbf{a})}{\eta + \omega} \right) \exp \left( \frac{-\eta - \omega - \lambda}{\eta + \omega} \right) \quad (47)$$

where the last term can be seen as a normalization constant

$$\exp \left( \frac{-\eta - \omega - \lambda}{\eta + \omega} \right) = \left( \sum_{\mathbf{a}} \pi_{\text{old}}(\mathbf{a}|s)^{\frac{\eta}{\eta+\omega}} \exp \left( \frac{\tilde{G}_{\mathbf{w}}^{\pi_{\text{old}}}(\mathbf{a}, s)}{\eta + \omega} \right) \right)^{-1}. \quad (48)$$

Plugging Eqs. (47) and (48) into Eq. (45) results in the dual [similar to Akrou et al. (2016)] used for optimization

$$L(\eta, \omega) = -\eta\epsilon - \omega\beta + \omega \sum_{s,a} p(s) H(\pi_{\text{old}}(a|s)) \\ - (\eta + \omega) \sum_s p(s) \log \left( \sum_a \pi_{\text{old}}(a|s)^{\frac{\eta}{\eta+\omega}} \exp \left( \frac{\tilde{G}_{\mathbf{w}}^{\pi_{\text{old}}}(a, s)}{\eta + \omega} \right) \right).$$

## B Technical details for discrete action experiments

Here, we provide details on the experiments with discrete actions. Table 3 shows details on the hyper-parameters used in the Field Vision RockSample (FVRS) experiments and Algorithm 2 describes details on the discrete action algorithm.

---

### Algorithm 2 COPOS discrete actions

---

Initialize policy network  $\pi_{\theta, \beta}$  with non-linear parameters  $\beta$  and linear parameters  $\theta$  and  $\Theta = (\theta, \beta)$   
**for** episode  $\leftarrow 1$  **to** maxEpisode **do**  
  Initialize empty batch  $\mathcal{B}$   
  **while** collected samples  $<$  batchsize **do**  
    Run policy  $\pi_{\theta, \beta}(a|s)$  for  $T$  timesteps or until termination: Draw action  $\mathbf{a}_t \sim \pi_{\theta, \beta}(\mathbf{a}_t|s_t)$ , observe reward  $r_t$   
    Add samples  $(s_t, \mathbf{a}_t, r_t)$  to  $\mathcal{B}$   
  **end while**  
  Compute advantage values  $A^{\pi_{\text{old}}}(s_i, \mathbf{a}_i)$   
  Compute  $\mathbf{w} = (\mathbf{w}_{\theta}, \mathbf{w}_{\beta})$  using conjugate gradient to solve

$$\mathbf{w} = \mathbf{F}^{-1} \nabla_{\Theta} J_{\text{PG}}(\pi_{\Theta})|_{\Theta=\Theta_{\text{old}}} \quad \nabla_{\Theta} J_{\text{PG}}(\pi_{\Theta}) = \sum_i^{|B|} \nabla_{\Theta} \log \pi_{\Theta}(\mathbf{a}_i|s_i) A^{\pi_{\text{old}}}(s_i, \mathbf{a}_i)$$

Use  $\tilde{G}_{\mathbf{w}}^{\pi_{\text{old}}}(s, \mathbf{a})$  to solve for  $\eta > 0$  and  $\omega > 0$  using the dual to the corresponding trust region optimization problem:

$$\begin{aligned} & \operatorname{argmax}_{\pi_{\theta}} \mathbb{E}_{s \sim p(s)} \left[ \int \pi_{\theta}(a|s) \tilde{G}_{\mathbf{w}}^{\pi_{\text{old}}}(s, a) da \right] \\ & \text{subject to } \mathbb{E}_{s \sim p(s)} [KL(\pi_{\theta}(\cdot|s) \parallel \pi_{\theta_{\text{old}}}(\cdot|s))] < \epsilon \\ & \quad \mathbb{E}_{s \sim p(s)} [H(\pi_{\theta}(\cdot|s)) - H(\pi_{\theta_{\text{old}}}(\cdot|s))] < \beta \end{aligned}$$

Apply updates for the new policy:

$$\theta_{\text{new}} = \frac{\eta \theta_{\text{old}} + \mathbf{w}_{\theta}}{\eta + \omega} \quad \beta_{\text{new}} = \beta_{\text{old}} + s \frac{\mathbf{w}_{\beta}}{\eta}$$

where  $s$  is a rescaling factor found by line search

**end for**

---

**Table 3** Parameters used for FVRS instances

	(5, 5) Full	(5, 5) Noise	(5, 7) Full	(5, 7) Noise	(7, 8) Full	(7, 8) Noise
Input space dim.	15	85	17	115	22	134
Output space dim.	5	5	5	5	5	5
# Policy parameters	1410	3510	1650	4560	1620	4980
Sim. step per iteration	5000	5000	5000	5000	5000	5000
Total num. of iterations	600	600	600	600	600	600
Discount $\gamma$	0.95	0.95	0.95	0.95	0.95	0.95
History length	1	15	1	15	1	15
Horizon	25	25	35	35	50	50

## C Additional continuous control experiments with a TRPO entropy bonus

Table 4 shows additional results for continuous control in the Roboschool environment. In these experiments, an additional entropy bonus is added to the reward function of TRPO.

**Table 4** Additional continuous control environment benchmark runs

	COPOS $\beta$ =auto	COPOS $\beta = \epsilon$	$\pi_1, \beta = 0.02$	$\pi_2, \beta = 0.02$
RoboInvDblPendulum-v1	7110.0 $\pm$ 416.0	7720.0 $\pm$ 235.0	8010.0 $\pm$ 58.0	6600.0 $\pm$ 330.0
RoboHopper-v1	<b>2510.0 <math>\pm</math> 35.1</b>	<b>2430.0 <math>\pm</math> 48.0</b>	1370.0 $\pm$ 37.1	703.0 $\pm$ 50.1
RoboWalker2d-v1	<b>1880.0 <math>\pm</math> 34.6</b>	<b>1910.0 <math>\pm</math> 41.5</b>	886.0 $\pm$ 33.3	69.3 $\pm$ 1.83
RoboHalfCheetah-v1	<b>2800.0 <math>\pm</math> 57.3</b>	<b>2770.0 <math>\pm</math> 42.2</b>	1500.0 $\pm$ 20.9	376.0 $\pm$ 42.6
RoboAnt-v1	<b>2330.0 <math>\pm</math> 49.7</b>	<b>2380.0 <math>\pm</math> 26.9</b>	1190.0 $\pm$ 41.0	614.0 $\pm$ 25.0
RoboHumanoid-v1	<b>114.0 <math>\pm</math> 0.937</b>	52.9 $\pm$ 0.306	20.2 $\pm$ 0.515	-7.29 $\pm$ 2.0
RoboHumanFlagrunHard-v1	<b>77.7 <math>\pm</math> 5.12</b>	32.1 $\pm$ 1.34	-121.0 $\pm$ 11.3	-55.4 $\pm$ 1.49
RoboAtlasForwardWalk-v1	<b>238.0 <math>\pm</math> 2.08</b>	187.0 $\pm$ 1.18	108.0 $\pm$ 0.825	73.9 $\pm$ 1.04
	$\pi_1, \beta = 0.01$	$\pi_2, \beta = 0.01$	$\pi_1, \beta = 0.005$	$\pi_2, \beta = 0.005$
RoboInvDblPendulum-v1	8030.0 $\pm$ 124.0	7630.0 $\pm$ 172.0	7730.0 $\pm$ 222.0	7780.0 $\pm$ 257.0
RoboHopper-v1	1620.0 $\pm$ 18.4	1530.0 $\pm$ 71.2	1940.0 $\pm$ 29.7	1930.0 $\pm$ 54.0
RoboWalker2d-v1	1140.0 $\pm$ 39.3	757.0 $\pm$ 91.0	1430.0 $\pm$ 25.5	939.0 $\pm$ 96.3
RoboHalfCheetah-v1	1840.0 $\pm$ 25.1	1250.0 $\pm$ 63.0	2180.0 $\pm$ 16.6	2190.0 $\pm$ 67.7
RoboAnt-v1	1930.0 $\pm$ 33.8	1910.0 $\pm$ 40.9	2220.0 $\pm$ 44.0	2130.0 $\pm$ 57.8
RoboHumanoid-v1	46.6 $\pm$ 2.03	56.5 $\pm$ 2.31	59.7 $\pm$ 3.99	95.3 $\pm$ 3.52
RoboHumanFlagrunHard-v1	-58.1 $\pm$ 2.37	-36.1 $\pm$ 1.96	-23.4 $\pm$ 1.93	-11.5 $\pm$ 1.78
RoboAtlasForwardWalk-v1	122.0 $\pm$ 0.692	87.2 $\pm$ 1.32	150.0 $\pm$ 0.538	136.0 $\pm$ 1.72
	$\pi_1, \beta = -0.02$	$\pi_2, \beta = -0.02$	$\pi_1, \beta = -0.01$	$\pi_2, \beta = -0.01$
RoboInvDblPendulum-v1	7760.0 $\pm$ 208.0	7670.0 $\pm$ 378.0	7340.0 $\pm$ 335.0	7490.0 $\pm$ 292.0
RoboHopper-v1	2200.0 $\pm$ 31.6	1800.0 $\pm$ 180.0	2130.0 $\pm$ 36.8	1950.0 $\pm$ 140.0

Table 4 continued

	$\pi_1, \beta = -0.02$	$\pi_2, \beta = -0.02$	$\pi_1, \beta = -0.01$	$\pi_2, \beta = -0.01$
RoboWalker2d-v1	1340.0 $\pm$ 148.0	653.0 $\pm$ 47.5	1610.0 $\pm$ 106.0	796.0 $\pm$ 45.0
RoboHalfCheetah-v1	2520.0 $\pm$ 20.9	1140.0 $\pm$ 215.0	2470.0 $\pm$ 40.0	1460.0 $\pm$ 243.0
RoboAnt-v1	<b>2320.0 <math>\pm</math> 38.4</b>	1340.0 $\pm$ 148.0	2270.0 $\pm$ 34.4	1790.0 $\pm$ 133.0
RoboHumanoid-v1	87.1 $\pm$ 3.36	98.1 $\pm$ 3.18	77.1 $\pm$ 2.09	106.0 $\pm$ 2.76
RoboHumanFlagrunHard-v1	<b>81.1 <math>\pm</math> 4.31</b>	36.8 $\pm$ 1.8	56.3 $\pm$ 1.5	37.1 $\pm$ 2.07
RoboAtlasForwardWalk-v1	199.0 $\pm$ 0.908	212.0 $\pm$ 4.19	195.0 $\pm$ 1.23	226.0 $\pm$ 2.34
	$\pi_1, \beta = -0.005$		$\pi_2, \beta = -0.005$	
RoboInvDbIPendulum-v1	7230.0 $\pm$ 348.0		7630.0 $\pm$ 265.0	
RoboHopper-v1	2130.0 $\pm$ 32.2		2040.0 $\pm$ 146.0	
RoboWalker2d-v1	1430.0 $\pm$ 153.0		962.0 $\pm$ 133.0	
RoboHalfCheetah-v1	2410.0 $\pm$ 17.0		1920.0 $\pm$ 240.0	
RoboAnt-v1	<b>2310.0 <math>\pm</math> 48.1</b>		2070.0 $\pm$ 33.1	
RoboHumanoid-v1	73.8 $\pm$ 2.66		106.0 $\pm$ 2.24	
RoboHumanFlagrunHard-v1	44.2 $\pm$ 1.85		27.4 $\pm$ 2.86	
RoboAtlasForwardWalk-v1	189.0 $\pm$ 1.08		221.0 $\pm$ 1.81	

In these experiments TRPO was run with an additional entropy term added to the reward function multiplied with a factor  $\beta$ . All algorithms are TRPO versions with the two different policy structures  $\pi_1(a|s)$ ,  $\pi_2(a|s)$  and different  $\beta$  except for the two COPOS entries. We report the mean of the average return over 50 last iterations  $\pm$  standard error over 10 random seeds. Bold denotes: no statistically significant difference to the best result (Welch's t-test with  $p < 0.05$ )

## References

- Abdolmaleki, A., Lioutikov, R., Peters, J., Lau, N., Reis, L., & Neumann, G. (2015). Model-based relative entropy stochastic search. In *Advances in Neural Information Processing Systems (NIPS)*, MIT Press.
- Abdolmaleki, A., Springenberg, J. T., Tassa, Y., Munos, R., Heess, N., & Riedmiller, M. (2018). Maximum a posteriori policy optimisation. In *Proceedings of the international conference on learning representations (ICLR)*.
- Akrour, R., Abdolmaleki, A., Abdulsamad, H., & Neumann, G. (2016). Model-free trajectory optimization for reinforcement learning. In *Proceedings of the international conference on machine learning (ICML)*.
- Akrour, R., Abdolmaleki, A., Abdulsamad, H., Peters, J., & Neumann, G. (2018). Model-free trajectory-based policy optimization with monotonic improvement. *Journal of Machine Learning Research*, 19(14), 1–25.
- Amari, S. (1998). Natural gradient works efficiently in learning. *Neural Computation*, 10(2), 251–276.
- Bagnell, J. A., & Schneider, J. (2003). Covariant policy search. *IJCAI*.
- Bernacchia, A., Lengyel, M., & Hennequin, G. (2018). Exact natural gradient in deep linear networks and its application to the nonlinear case. In *Advances in Neural Information Processing Systems (NIPS)*, Curran Associates, Inc., pp 5945–5954.
- Boyd, S., & Vandenberghe, L. (2004). *Convex optimization*. Cambridge: Cambridge University Press.
- Daniel, C., Neumann, G., Kroemer, O., & Peters, J. (2016). Hierarchical relative entropy policy search. *Journal of Machine Learning Research (JMLR)*, 17(93), 1–50.
- Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., & Koltun, V. (2017). CARLA: An open urban driving simulator. In *Conference on robot learning*, pp. 1–16.
- Duan, Y., Chen, X., Houthoofd, R., Schulman, J., & Abbeel, P. (2016). Benchmarking deep reinforcement learning for continuous control. In *Proceedings of the 33rd international conference on machine learning, ICML 2016*, New York City, NY, USA, June 19–24, 2016, pp 1329–1338. <http://jmlr.org/proceedings/papers/v48/duan16.html>.
- Geist, M., & Pietquin, O. (2010). Revisiting natural actor-critics with value function approximation. In *International conference on modeling decisions for artificial intelligence*, Springer, pp. 207–218.

- Hansen, N., & Ostermeier, A. (2001). Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2), 159–195.
- Kakade, S. (2001). A natural policy gradient. In T. G. Dietterich, S. Becker, & Z. Ghahramani (Eds.), *Advances in neural information processing systems 14 (NIPS 2001)* (pp. 1531–1538). Cambridge: MIT Press.
- Kober, J., & Peters, J. R. (2009). Policy search for motor primitives in robotics. In D. Koller, D. Schuurmans, Y. Bengio, & L. Bottou (Eds.), *Advances in neural information processing systems 21* (pp. 849–856). Red Hook: Curran Associates, Inc.
- Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., & Wierstra, D. (2015). Continuous control with deep reinforcement learning. [arXiv:1509.02971](https://arxiv.org/abs/1509.02971).
- Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., et al. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540), 529–533.
- Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., & Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pp. 1928–1937.
- O'Donoghue, B., Munos, R., Kavukcuoglu, K., & Mnih, V. (2016). PGQ: Combining policy gradient and q-learning. [arXiv:1611.01626](https://arxiv.org/abs/1611.01626).
- Peters, J., & Schaal, S. (2008). Natural actor-critic. *Neurocomputing*, 71(7–9), 1180–1190.
- Peters, J., Mülling, K., & Altun, Y. (2010). Relative entropy policy search. In *AAAI Atlanta*, pp. 1607–1612.
- Ross, S., Pineau, J., Paquet, S., & Chaib-Draa, B. (2008). Online planning algorithms for POMDPs. *Journal of Artificial Intelligence Research*, 32, 663–704.
- Rubinstein, R. (1999). The cross-entropy method for combinatorial and continuous optimization. *Methodology and Computing in Applied Probability*, 1(2), 127–190.
- Schulman, J., Levine, S., Abbeel, P., Jordan, M., & Moritz, P. (2015). Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pp. 1889–1897.
- Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. [arXiv:1707.06347](https://arxiv.org/abs/1707.06347).
- Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., & Riedmiller, M. (2014). Deterministic policy gradient algorithms. In *ICML*.
- Sutton, R. S., McAllester, D., Singh, S., & Mansour, Y. (1999). Policy gradient methods for reinforcement learning with function approximation. In *Proceedings of the 12th international conference on neural information processing systems*, MIT Press, Cambridge, MA, USA, NIPS'99, pp. 1057–1063.
- Tangkaratt, V., Abdolmaleki, A., & Sugiyama, M. (2018). Guide Actor-Critic for Continuous Control. In *Proceedings of the international conference on learning representations (ICLR)*.
- Wierstra, D., Schaul, T., Peters, J., & Schmidhuber, J. (2008). Natural evolution strategies. In *IEEE congress on evolutionary computation*, IEEE, pp 3381–3387.
- Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3–4), 229–256.
- Wu, Y., Mansimov, E., Grosse, R. B., Liao, S., & Ba, J. (2017). Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation. In *Advances in neural information processing systems (NIPS)*, pp. 5279–5288.